Efficient Data Cleaning Algorithm and Swift Unique User Identification Algorithm Using Coalesced Hashing and Binary Search Techniques for Web Usage Mining

1Ranjena Sriram 2S. Sheeja and 3I. Henry Alexander
1Karpagam University,
Karpagam Academy of Higher Education,
Coimbatore, Tamil Nadu, India.
ranjenasriram@gmail.com
2Department of Computer Applications,
Karpagam University,
Karpagam Academy of Higher Education,
Coimbatore, Tamil Nadu, India.
sheejaaajize@gmail.com
3Udumalpet,
Tirupur District, Tamil Nadu, India.
henryalexander695@gmail.com

Abstract

The overall study focuses on proposing a new Data cleaning and Unique User Identification processing and Unique User Identification algorithms for Web Usage Mining to discover and analyse the user’s access pattern through mining of log files or log databases and the associated data from a particular website. Pre-Processing technique is to clean the data and user identification process to identify unique users. Since number of users interacting with web sites around the world is increasing day by day, the amount of data generated and information gathered could help the organizations to improve their business according to the customers’ needs and behavior.

Web Log Server the main source of data has to organize the huge volume of user data and accordingly the searching has to be done efficiently. To overcome these problems a innovative Hashing technique
proposed in earlier studies, proved its performance in terms of accuracy and efficiency, but lacked in terms of collision and storage. Generally hash structures need more memory and often data try to collide with the memory location already occupied by other data (collision). Considering these issues this paper has come out with innovative Hashing strategies to minimize storage and avoid collision. Data collected from the Web Servers of Murdoch University, Emirates College for Management and Information Technology from United Arab Emirates and Nehru Arts and Science College from India are taken for evaluating the performance of modified Hashing Function implemented in the proposed Unique User Identification Algorithm. This study has contributed promising results in terms of storage, efficiency and accuracy over the previous studies.
1. Introduction

Internet has become an important source of information for all the types of users across the globe. Today much emphasis is given to the data used in web applications to process meaningful information. Most of the business people who use Web applications to promote their business, self-evaluate their websites to improve their business. To overcome this much preprocessing has to be done to the data in the Web Log Server to generate meaning patterns for processing, which guides the developers to focus on the grey areas and further directs them to end in a complete, full-fledged web site. An efficient data cleaning technique and a unique user identification algorithm are proposed. The proposed Unique User Identification algorithm is implemented with the help of proposed Hashing techniques blended with Binary Search strategies. These earlier studies of Hashing techniques consumed more storage and lacked in collision problems. This study has come out with innovative methodologies in Hashing techniques to overcome these problems.

2. Web Usage Mining

Web Usage mining is a Data mining used to discover the usage patterns from the Web to understand the better needs of Web applications. It extracts information from surfer’s session for further processing. Web content and web structure make use of the primary data while Web Usage mining use the secondary data from the Web Log files stored in the Web Servers. [1]

Web Usage Mining is a three phase process consisting:

a) Preprocessing / Data Preparation – Meaningful and proper dataset are needed to derive meaningful patterns. Getting relevant data from the various sources in the present world is a major task. Improper and irrelevant dataset will result to inaccurate and poor results. Studies are still in progress for overcome this problem. Hence much emphasis is given to this phase to get accurate data. Data preprocessing is the most difficult task in the mining process. Efficient, robust and versatile algorithms from other domains are used in this process to derive meaningful patterns.

b) Pattern Discovery: Patterns are generated using the Statistical Data mining, Associate rules and Sequential methods. These strategies make classification fast and efficient. Statistical methods, Data mining methods, Associate rule, Sequential methods and cluster techniques are used to identify unique patterns. Unique patterns derived from implementing these strategies helps to derive accurate and efficient results after mining.

c) Pattern Analysis: The patterns generated are analyzed using the OLAP tools, query management and smart agent based systems to remove irrelevant data, rules or patterns. This process improves the accuracy of the data, which further results in accurate mining[1].
3. Sources and Types of Data

The major data source for the Web Usage Mining is the server log files, which include web server access and application server logs. Apart from this information, additional data sources are essential for the data preparation and pattern discovery, which include site files, Meta data, operational databases, application template and vast domain knowledge. In some cases data from client side, proxy level data collection (Internet Service Provider), and demographic data sources provide by the data aggregation services are used for huge mining systems\cite{1}\cite{2}.

The data from the various sources can be categorized into four primary groups.

a) **Usage Data**: Web server serves as the main source of data for Web Usage mining. Each log record stored in the Web Server after each hit on the server by the users HTTP request, contains useful information like users IP address, time of request, information about various agents like users browser information, users operating systems and cache details. This primary information helps to mine meaningful patterns, which guides the Web developers to further improve their sites.

b) **Content Data**: Collection of objects and relations are conveyed to the users. Most of the data are in the format of text, images and structures generated from HTML and XML pages. Multimedia files dynamically generate page segments from scripts and record collection from the databases, meta data, document attributes, and HTML variables. Domain oncology such as content and relationship via oncology language such as RDF or a database schema over the data contained in the operational database.

c) **User Data**: Data collected from the operational databases which include demographic information about registered users, user ratings on various objects such as product or movies, past purchases or visit histories of users as well as other explicit or implicit representations of user’s interest.
4. Previous Work

An efficient Data Cleaning Algorithm along with an innovative Unique User Identification Algorithm was proposed in previous study work by inclusion of some additional factors and there by proved the improved performance of the algorithm by evaluating it with different web log files from web servers.

1. Data Cleaning

The principle of Data Cleaning is removed or reduces extraneous data. The following data are removed.

- Records containing video, graphics and file extensions of GIF, JPEG and CSS.
- The log records with the status codes over 299 or fewer than 200.
- Records having value of POST or HEAD.
- User agents like Crawler, Spider or Robot.

2. Proposed Algorithm for Data Cleaning

The generalized approach followed by the earlier strategies cannot apply in real time scenarios, which are handling huge volume of data and where time is an important criterion. Considering these prevailing conditions this study introduces an innovative algorithm which follows Generalized Pattern Sequence methodology to check for irrelevant data for Web Usage Mining. Since Generalized Pattern Sequence is a mining algorithm, only the salient features are extracted for pre-processing technique.

Individual groups for file extensions, server request and response methods, web site status and user agents are created. Accordingly each log record is fragmented and the above mentioned information is extracted from them. The fragments are simultaneously compared with the groups, if either one matches then the record is invalid or considered as irrelevant record and can be eliminated. The algorithm is explained below in detail.

3. Data Cleaning Algorithm using Generalized Pattern Sequence Methodology Proposed in Previous Work

Three sequences taken for the algorithm

- File extensions like (css, jpeg, jpg, js, gif)
- Methods (GET, POST)
- Site Status (301,404,500)
- User Agents.

Input: Web server Log File
Step 1: Let F be the different Groups
k =2
Step 2: Read Log Record from Web Server Log File
Step 3: Fragment Log Record into different elements fr.
Step 4: Do while (F k,t != Group Count)
Step 5: Let (a) denote individual fragments in Group F k
For all input fragments from Log Record \( r \) in Log file (or) Database D
Step 6: If \( a \) matches \( fr \) then
Step 7: Move the record to the corresponding Group and eliminate the record from Log Database
Else move to next group
\[ k = k + 1 \]
else
Consider the fragment as outlier.
End if
Step 8: Repeat until eof
End do

**Execution of the Algorithm**
- Input Log Record from log File
- Generate different Groups
- Read Log Record from the Log File and repeat until end of file.
- Fragment the Log Record into individual elements
- Compare each element in the groups with the input element from Log File
- If matches move the element to the individual group else move to next group.
- Eliminate the record from Log File
- Repeat the process until all groups are visited.

**4. Advantages**
- Searching time minimizes since the given element from the log record is parallel checked in all groups.
- Efficient and quick when comparing with other techniques.

**5. User Identification**

User’s identification is, to categorize who access web site and which pages are accessed. Different users may have same IP address in the log. A referrer-based method is proposed to solve these problems in this study.

The rules adopted to distinguish user sessions can be described as follows:
- Each IP address represents one user.
- For more logs, if the IP address is the same, but the agent log shows a change in browser software or operating system, an IP address represents a different user.
- Using the access log in conjunction with the referrer logs and site topology to construct browsing paths for each user. If a page is requested that is not directly reachable by a hyperlink from any of the pages visited by the user, there is another user with the same IP address.

This work comes out with an innovative Unique User Identification algorithm using Hashing techniques to organize and locate the user in quick manner, though it is efficient it has its own drawbacks which is modified and proposed.
as modified algorithm which uses grouping of the similar zone IP’s followed by Hashing and Binary Search techniques to locate the user more faster when comparing with this UUI algorithm.

1. **Proposed Unique User Identification Algorithm Using Hashing Technique**

Unique user identification is an important process next to data cleaning. Unique users are identified based on the rules suggested in the User Identification section. Though many efficient algorithms are there, many fail in accuracy and efficiency (time taken to identify users) when the size of the Log Database increases.

Today’s modern web servers are capable of handling terabytes of data. Conventional algorithms are obsolete in handling these scenarios. Considering the above facts, this study proposes an efficient Unique User Identification algorithm that uses modern Hashing techniques to identify unique user quickly inspire the huge size of the database.

A new hashing key is prosed and successfully implemented in the algorithm to locate the user [6].

**Hashing Techniques**

For a huge database structure, it can be almost next to impossible to organize and search all the index values through all its level and then reach the destination data block to retrieve the desired data. Hashing is an effective technique to calculate the direct location of a data record on the disk without using index structure both for storage and retrieval of data.

Hashing uses hash functions with search keys as parameters to generate the address of a data record [5].

**Hash Organization**

Generally a hash store the data in the form of a bucket, a bucket is a representative of a storage block which stores one complete disk block, which in turn stores record groups. Searching in Hash table is done by a Hash Function which maps all set of search keys (K) to the address where the actual records are placed. It is a function from search key to bucket addresses [5].

**Dynamic Hashing**

The problem with static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks. Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically on-demand. Dynamic hashing is also known as extended hashing [6].

Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially.
Generally any Unique User Identification algorithm analyses more factors like users IP addresses, web site topology, browser edition and operating system. The proposed algorithm not only uses the IP addresses but also identifies the user’s session. The proposed algorithm not only uses User IP address, but also based on path chosen by any user, access time with the referred page etc.\cite{4}.

When huge databases are taken for considering the time taken to locate the records is much, hence an appropriate methodology is incorporated to make the process faster. Taking these prevailing conditions, the study proposes a new Hashing formulation, to minimize the searching time for large datasets. Previous study proposed a Hashing function, which is quick enough to search the unique user’s IP address, but when the size of the bucket increases certain pre-processing is done to fasten the searching time of the unique user. On considering the prevailing issues, this work substitutes Binary search techniques to minimize the searching time.

A few modifications are done in previously proposed Hashing function to make it generalized and quick in searching patterns.

**Hash Function Proposed in Previous Work**

\[ N \text{mod} 10 \times K + d \quad (1.1) \]

Where \( N \) denotes the sum of users IP address, \( (K) \) refers to the virtual address of the bucket and \( d \) refers to the displacement distance. The multiplied factor gives the original location of the data\cite{6}.

**Example consider an IP 192.168.1.1**

The sum of IP address is \( 192 + 168 + 1 + 1 = 132 \rightarrow N \)

According to the Hash function \( 132 \text{mod} 10 = 2 \)

Hence the values are stored in the index value of 2.

Substitute \( N \text{mod} 10 \) with parameter \( H \) equation (1.1) becomes

\[ H (K) \quad (1.2) \]

An error function was introduced in further studies for accurate and efficient
retrieval if data. The inclusion of error factor further improved the performance of the Unique User Identification Algorithm.

Now equation 1.1 is transformed to

\[ N_{\text{mod}10} * K + d + \Delta E \quad (1.3) \]

The generalized function by inclusion of error factor is

\[ H(K) + \Delta E. \quad (1.4) \]

Where \( \Delta E \) is the new error factor

\[ \Delta E = \text{Original memory address} - \text{Address obtained from Hashing function.} \]

The error function removes the discrepancy obtained from the hash function and helps to obtain the exact address required to fetch the user record.

**Drawbacks**

- Takes more time to locate the required user.
- Some pre-processing required minimizing the searching time.
- Takes user records (IP addresses) as such without splitting zone wise hence more time is needed identifying users.
- Consumes more storage if entire Hash table is taken for consideration.
- No provision was given to handle collision problem.

Considering these drawbacks Coalesced Hashing techniques are used in this study to overcome storage and collision problems.

**Coalesced Hashing**

Is a Hashing technique to remove collision and forms a hybrid of separate chaining and open addressing. In this technique hashing buckets are organized in form of linearly linked list structures. Node containing two values, the data and the address to the next location. Normally the hashing functions resulting in same key value are placed as neighboring nodes.

![Coalesced Hashing Diagram](image-url)
Figure 3(b): Coalesced Hashing

Figure-3(a) and 3(b) illustrates the organizing strategy of the data according to Coalesced Hashing technique. It is evident that the joining nodes as illustrated in the figures have unique hashing key values.

**Drawbacks of Open Hashing**
- Extra memory usage.
- Takes long time to search long sequences of used buckets that contains items with different hashing addresses
- Items with one hash address can thus lengthen searches for items of different hashing addresses.

Coalesced hashing removes these drawbacks by implementing certain strategies. Instead of creating new nodes for new data, buckets from the same Hash table are used. A bucket is allocated to store the data having similar hashing key value, and a link is established between this bucket and the existing chain. Thus it is possible to insert a new data with a different hashing address to merge with the items having different hashing addresses. Later the data stored in this bucket is coalesced with nodes which have similar hashing value. Figure (4) illustrates a Hash bucket with a NULL value, which is used to store the data which collides with the address of other data. From Figure (3) it can be e that keys of data 98, 10, 20 and 100 collide with each other. According to Coalesced Hashing technique data 10 is first stored in the Hash bucket having NULL value, later this node is placed next to the node containing the value 98, similar procedure is followed for data 20 and 100. Thus this strategy removes collision problem and minimizes the storage to a great extent [4][5][7].

An important optimization, to reduce the effect of coalescing, is to reduce the address space of the hashing function to the subset of the table. For example consider the table of size M with hash buckets ranging from 0 to M-1, address space can be restricted to first N locations in the table. Remaining M-N buckets called cellar are allocated for the data resulting in collision. Collision will not occur until the data in cellar are exhausted.
Procedure to Insert a New Node
N - Allotted memory locations
M - Size of the Hash table.
Htab – Hash table
int insert (char key[])
{
    PHASE-I :
    unsigned  h = hash(key, strlen(key)) % N
    //New hash function is generated and creates a new node if the hash bucket is NULL
    if (htab[h] == NULL)
    {
        htab[h]= make_node(key, NULL)
    }
    else
    {
        PHASE-II:
        //The hash key has collision hence inserted in (M-N) cellar bucket.
        struct node *lt
        int cursor = M-1
        while (cursor >0 && htab[cursor] != NULL)
        {--cursor
         
        }
        if (cursor==-1)
        {
            return -1
        }
        else{
            htab[cursor]= make_node(key, NULL)
            it = htab[h]
            while (it ->next !=NULL)
            {
                it=it->next
                it->next=htab[cursor]
            }
        }
    }
    return 0
}

Execution of the Algorithm
1. In PHASE-I a new node os created for the new Hash key if the Hash Table has enough locations to occupy the new value
2. Else in PHASE-II the data for the corresponding Hash key is inserted in the cellar buckets (i.e) (M-N) locations to avoid collision.
3. Further values are added to theses slots in case of collision only when theses slots are exhausted.

**Advantages of the new Hashing Function**
1) Prevents collision problems in Hashing  
2) Minimizes the storage.  
3) Accurately locates and retrieves the data from the memory location.  
4) Increases the accuracy and efficiency.  
5) Improves the overall efficiency of the Unique User Identification Algorithm

2. **Proposed Unique User Identification Algorithm**

Web Log server contains accumulated log information, which makes the searching more complicated. In order to minimize the searching time this work includes two main strategies.

- An algorithm is designed and developed to group IP addresses of similar zones in the individual Hash buckets from the Web Log Server file, which minimizes the searching time to a great extent.
- Binary Search techniques are used along with some string manipulations in the previously proposed UUI algorithm to reduce the searching time.

Different sets of IP ranges are allocated to particular networks, geographic areas, companies etc. The table below shows the several examples of IP ranges and their implementations.

3. **First Strategy**

<table>
<thead>
<tr>
<th>Table 1: IP Addresses of Different Zones</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Range</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>10.16.121</td>
</tr>
<tr>
<td>192.168</td>
</tr>
<tr>
<td>255</td>
</tr>
<tr>
<td>91</td>
</tr>
</tbody>
</table>

Since different zones start with different IP address and the IP address of the given user is searched must be differentiated and grouped to their specific zones in the separate Hash buckets assigned for different IP zones from other zones of IP addresses. The algorithm explained below shows how specific IP address from a particular zone is extracted and stored in the separate Hash bucket in the form of array. This tactic reduces the searching time to much extent which is shown in the result and discussions section[10][11].
Set Ip Algorithm to Group IP Zone Wise from Web Log Server File

4. Execution of Set Ip Algorithm
   • Read records one by one until the end of file.
   • Get first three digits from each record.
   • Match with the case statement.
   • If the record matches to the case statement, store the IP in the arrays assigned to each zone.

5. Second Strategy

Now this arrangement facilitates the modified UUI algorithm to search whether
the IP of the user exists or not quickly. In order to achieve this task some modifications are done to the previously proposed UUI algorithm.

- Binary Search techniques are used along with some string manipulations in the previously proposed UUI algorithm to reduce the searching time.
- The Binary Search Algorithm is combined with the proposed Hashing function to locate the address of the data.

Figure 6: Unique User Identification (UUI) Algorithm

6. Execution of the Algorithm

- Get the input user IP.
- Extract the first three digits from it.
- Check to which zone it belongs using the switch statement.
- If it matches a particular zone, then search the given user IP in that particular zone Hash bucket using Binary Search and Hash function.
- If found extract the user information, if not assign the IP address to that particular zone and treat it as new user.
7. **Advantages of the Modified UUI Algorithm**

- Since the IP addresses are grouped zone wise, easy to search and locate the users IP addresses and their relevant information.
- Binary Search techniques are combined with Hash function which makes the searching faster minimizing the time.
- This proposed algorithm proves and shows better results over other UUI algorithms, which are elaborated in the results and discussions section.

6. **Results and Discussions**

To validate the effectiveness and efficiency of the algorithms proposed, an experiment with the web server logs of Murdoch University and Emirates College of Management and Information Technology, Dubai and Nehru Arts and Science College, India, is made. This work has proposed a effective Hashing technique, which minimizes the storage and eradicates collision problem. These changes have drastically improved the searching time of the user record and thereby improve its performance. Results obtained from the previous work are compared with the updated work. This work proves with better results to validate the work done. The initial data source of our experiment is from JAN 1, 2014 to Aug 3, 2015, with data size of 10^{12} records. The experiments are per-formed on a 2.8GHz Intel Celeron I, CPU, 2.00 GB of main memory, Windows 2000 professional, SQL Server 2000 and MATLAB (7.9.0.529). MATLAB tool is used to develop applications to evaluate the performance of the proposed algorithms. The table listed below illustrates the overall performance of UUI algorithm.

<table>
<thead>
<tr>
<th>Data Sources</th>
<th>Record Size</th>
<th>No of Collisions from Previous Hashing Function</th>
<th>No of Collisions from Modified Hashing function using Coalesced technique</th>
<th>Memory consumed for previous Hashing Technique</th>
<th>Memory consumed for Modified Hashing Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>MURDOC UNIVERSITY</td>
<td>100000279900 (records)</td>
<td>1027</td>
<td>12</td>
<td>10.236GB</td>
<td>6.286GB</td>
</tr>
<tr>
<td>EMIRATES COLLEGE OF MANAGEMENT AND INFORMATION TECHNOLOGY</td>
<td>100450279900 (records)</td>
<td>1467</td>
<td>08</td>
<td>12.124GB</td>
<td>8.472GB</td>
</tr>
<tr>
<td>NEHRU ARTS AND SCIENCE COLLEGE</td>
<td>125232787204 (records)</td>
<td>1259</td>
<td>14</td>
<td>11.335GB</td>
<td>5.5462GB</td>
</tr>
</tbody>
</table>

From Table (2) it is evident that number of collisions by the user records for
occupying the memory locations is minimized completely by implementing the modified Coalesced Hashing Technique. From the above figure it is evident that the storage for the user records is minimized to great extent over the previous study.

Table 3: Comparison Results of Data Cleaning and Unique User Identification Process from Previous Work with Murdoc University, ECMIT College and Nehru Arts and Science College

<table>
<thead>
<tr>
<th>DATA SOURCES</th>
<th>PREVIOUS STUDY</th>
<th>MODIFIED STUDY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entries in raw web log</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entries after data cleaning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Unique users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution time of UUI(Algorithm)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of sessions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the Table(3) it is clearly evident that the modified algorithm works fine for a large dataset and also proves the improved performance over the previous work in terms of accuracy and efficiency. Results prove that the proposed UUI
algorithm consumes relatively less time to find whether the user record already exists or a new one. The following sections describe the performance of the proposed UUI Algorithm with Murdoch University, Dubai server log data.

Table 4: Overall Performance of Proposed UUI Algorithm’s for Murdoch University

<table>
<thead>
<tr>
<th>Month</th>
<th>Unique Visitor(s)</th>
<th>Number of Visits</th>
<th>Pages</th>
<th>Hits</th>
<th>Unique Visitors</th>
<th>Number of Visits</th>
<th>Pages</th>
<th>Hits</th>
<th>Bandwidth from (First Study)</th>
<th>Bandwidth from (Second Study)</th>
<th>Bandwidth from (Third Study)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 2014</td>
<td>747792</td>
<td>371</td>
<td>94759</td>
<td>2371</td>
<td>194759</td>
<td>2371</td>
<td>194759</td>
<td>2371</td>
<td>1.9 GB</td>
<td>1.782 GB</td>
<td>1.526 GB</td>
</tr>
<tr>
<td>Feb 2014</td>
<td>726527</td>
<td>342</td>
<td>83652</td>
<td>7342</td>
<td>283627</td>
<td>342</td>
<td>172652</td>
<td>7342</td>
<td>1.85 GB</td>
<td>1.653 GB</td>
<td>1.435 GB</td>
</tr>
<tr>
<td>Mar 2014</td>
<td>718945</td>
<td>720</td>
<td>171894</td>
<td>5720</td>
<td>171894</td>
<td>720</td>
<td>171894</td>
<td>5720</td>
<td>1.65 GB</td>
<td>1.427 GB</td>
<td>1.257 GB</td>
</tr>
<tr>
<td>Apr 2014</td>
<td>727654</td>
<td>381</td>
<td>181765</td>
<td>4381</td>
<td>181765</td>
<td>381</td>
<td>181765</td>
<td>4381</td>
<td>1.7 GB</td>
<td>1.65 GB</td>
<td>1.42 GB</td>
</tr>
<tr>
<td>May 2014</td>
<td>625678</td>
<td>990</td>
<td>655678</td>
<td>8990</td>
<td>1755678</td>
<td>990</td>
<td>1755678</td>
<td>990</td>
<td>1.54 GB</td>
<td>1.35 GB</td>
<td>1.15 GB</td>
</tr>
<tr>
<td>June 2014</td>
<td>543298</td>
<td>760</td>
<td>643298</td>
<td>8760</td>
<td>1643298</td>
<td>760</td>
<td>1643298</td>
<td>8760</td>
<td>1.22 GB</td>
<td>1.02 GB</td>
<td>0.98 GB</td>
</tr>
<tr>
<td>July 2014</td>
<td>456789</td>
<td>321</td>
<td>565789</td>
<td>9321</td>
<td>1656789</td>
<td>321</td>
<td>1656789</td>
<td>9321</td>
<td>1.02 GB</td>
<td>0.98 GB</td>
<td>0.85 GB</td>
</tr>
<tr>
<td>Aug 2014</td>
<td>326789</td>
<td>900</td>
<td>526789</td>
<td>9900</td>
<td>1526789</td>
<td>900</td>
<td>1526789</td>
<td>900</td>
<td>1.00 GB</td>
<td>0.75 GB</td>
<td>0.65 GB</td>
</tr>
<tr>
<td>Sep 2014</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Oct 2014</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Nov 2014</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dec 2014</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

From Table (4) the results prove the improved performance of the modified study over the previous studies. The decrease in bandwidth over the previous studies substantiates the improved efficiency of the Modified UUI Algorithm along with the updated Hashing techniques for Murdoch University Web data.
Table 5: Overall Performance of Proposed UUI Algorithm’s for Nehru Arts and Science College

<table>
<thead>
<tr>
<th>Month</th>
<th>Visitor Unique</th>
<th>Unique</th>
<th>Uniqu</th>
<th>Pages</th>
<th>Hits</th>
<th>Unique</th>
<th>Pages</th>
<th>Hits</th>
<th>Band width obtained from (First Study)</th>
<th>Band width obtained from (Seconed Study)</th>
<th>Band width obtained from (Third Study)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREVIOUS STUDY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jan 2015</td>
<td>83478 367</td>
<td>847592 352</td>
<td>936592 378</td>
<td>296759 2371</td>
<td>734784 367</td>
<td>947592 352</td>
<td>946592 378</td>
<td>1396759 2371</td>
<td>1.923 GB</td>
<td>1.8 GB</td>
<td>1.75 GB</td>
</tr>
<tr>
<td>Feb 2015</td>
<td>825378 634</td>
<td>836527 242</td>
<td>836527 342</td>
<td>283652 7342</td>
<td>725378 634</td>
<td>836527 242</td>
<td>946527 342</td>
<td>1283652 7342</td>
<td>1.852 GB</td>
<td>1.72 GB</td>
<td>1.64 GB</td>
</tr>
<tr>
<td>Mar 2015</td>
<td>813975 254</td>
<td>818945 566</td>
<td>818945 720</td>
<td>251894 5720</td>
<td>713975 254</td>
<td>818945 566</td>
<td>918945 720</td>
<td>1251894 5720</td>
<td>1.652 GB</td>
<td>1.50 GB</td>
<td>1.25 GB</td>
</tr>
<tr>
<td>Apr 2015</td>
<td>812745 238</td>
<td>817654 631</td>
<td>847654 381</td>
<td>192765 4381</td>
<td>712745 238</td>
<td>817654 631</td>
<td>957654 381</td>
<td>1927654 381</td>
<td>1.723 GB</td>
<td>1.60 GB</td>
<td>1.32 GB</td>
</tr>
<tr>
<td>May 2015</td>
<td>825678 990</td>
<td>855678 990</td>
<td>855678 990</td>
<td>187767 8990</td>
<td>715678 990</td>
<td>855678 991</td>
<td>955678 990</td>
<td>1877678 990</td>
<td>1.541 GB</td>
<td>1.30 GB</td>
<td>1.10 GB</td>
</tr>
<tr>
<td>June 2015</td>
<td>643527 876</td>
<td>643298 760</td>
<td>843298 8760</td>
<td>176329 8760</td>
<td>543527 876</td>
<td>643298 760</td>
<td>943298 760</td>
<td>1763298 760</td>
<td>1.221 GB</td>
<td>1.02 GB</td>
<td>0.96 GB</td>
</tr>
<tr>
<td>July 2015</td>
<td>843278 546</td>
<td>656789 321</td>
<td>756789 321</td>
<td>263378 9321</td>
<td>643278 546</td>
<td>656789 321</td>
<td>956789 321</td>
<td>2633789 321</td>
<td>1.024 GB</td>
<td>0.95GB</td>
<td>0.75 GB</td>
</tr>
<tr>
<td>Aug 2015</td>
<td>826578 387</td>
<td>826789 900</td>
<td>862678 900</td>
<td>252678 9000</td>
<td>726578 387</td>
<td>826789 900</td>
<td>962678 9000</td>
<td>1252678 9900</td>
<td>1.001 GB</td>
<td>0.95GB</td>
<td>0.65 GB</td>
</tr>
<tr>
<td>Sep 2015</td>
<td>765342 781</td>
<td>847592 352</td>
<td>866378 990</td>
<td>254359 2371</td>
<td>665342 781</td>
<td>847592 352</td>
<td>986367 890</td>
<td>1254359 2371</td>
<td>1.632 GB</td>
<td>1.50 GB</td>
<td>1.30 GB</td>
</tr>
<tr>
<td>Oct 2015</td>
<td>853452 678</td>
<td>836527 242</td>
<td>858398 760</td>
<td>264785 7342</td>
<td>753452 678</td>
<td>836527 242</td>
<td>958539 8760</td>
<td>1264785 7342</td>
<td>1.041 GB</td>
<td>0.92 GB</td>
<td>0.67 GB</td>
</tr>
<tr>
<td>Nov 2015</td>
<td>853567 846</td>
<td>818945 566</td>
<td>856788 446</td>
<td>253837 5720</td>
<td>653567 846</td>
<td>818945 566</td>
<td>985678 8446</td>
<td>1258372 5720</td>
<td>1.63 GB</td>
<td>1.52 GB</td>
<td>1.21 GB</td>
</tr>
<tr>
<td>Dec 2015</td>
<td>765432 876</td>
<td>817654 631</td>
<td>847654 381</td>
<td>292765 4381</td>
<td>665432 876</td>
<td>817654 631</td>
<td>984765 4381</td>
<td>1292765 4381</td>
<td>2.00 GB</td>
<td>1.35 GB</td>
<td>1.37 GB</td>
</tr>
<tr>
<td>Total</td>
<td>966385 3473</td>
<td>962399 6554</td>
<td>995149 7369</td>
<td>297551 06599</td>
<td>205638 33473</td>
<td>207239 6654</td>
<td>211514 97369</td>
<td>1599736 64870</td>
<td>18.18 GB</td>
<td>16.33 GB</td>
<td>13.97 GB</td>
</tr>
</tbody>
</table>

From Table(5) it is clearly proved that the modified UUI Algorithm along with the modified Coalesced Hashing technique. The decreased bandwidth occupied by the modified UUI Algorithm proves the improved performance over the previous contributions for Nehru Arts and Science College Web data.
The graphical results displayed in the Figures 7(a) and 7(b) illustrate the results of the older version of UUI algorithm over the modified version for the data collected from Murdoch University web server. From the results obtained it is evident that the modified UUI shows better result in terms of efficiency, accuracy and decrease in bandwidth over the older version. Similarly the Figures 8(a) and 8 (b) shows the comparison of the performance of older and new versions of UUI algorithms. From the results displayed, it is evident that the proposed Data cleaning algorithm performs well along with the proposed UUI algorithm with modified hash function for huge Web Log Server data.

Similarly Figures 8(a) and 8(b) illustrates the improved performance of the proposed UUI Algorithm implemented with coalesced hashing techniques over
the previous study. From the graphical displays it is evident that the bandwidth consumed by modified UUI Algorithm is less than the previous study. The less storage consumed by the coalesced hashing technique to store the data, improvises the performance of the modified UUI Algorithm by faster retrieval of data.

Table 6: Performance of Modified UUI Algorithm with other Related Works

<table>
<thead>
<tr>
<th>Performance Analysis</th>
<th>Database Source</th>
<th>Record Size</th>
<th>Entries in Raw Web Log</th>
<th>Entries After Data Cleaning</th>
<th>No of Users</th>
<th>No of Unique Users</th>
<th>Execution Time(s) Previous Work</th>
<th>Execution Time(s) Modified Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Modified Unique User Identification Algorithm</td>
<td>Web Server Log of University</td>
<td>10^7</td>
<td>1000000027900 0</td>
<td>10000000278 3</td>
<td>56750287 6</td>
<td>43667542</td>
<td>0.4247</td>
<td>0.3582</td>
</tr>
<tr>
<td>Proposed Modified Unique User Identification Algorithm</td>
<td>NASA Server Log</td>
<td>10^7</td>
<td>87233</td>
<td>33657</td>
<td>4000</td>
<td>1765</td>
<td>0.5432</td>
<td>NIL</td>
</tr>
<tr>
<td>Proposed Modified Unique User Identification Algorithm</td>
<td>Web Server Log of Nehru Arts and Science College, India</td>
<td>10^7</td>
<td>12564400027 7</td>
<td>1124000277 1</td>
<td>57750287 6</td>
<td>44667542</td>
<td>0.4211</td>
<td>0.3724</td>
</tr>
</tbody>
</table>

Figure 9: Number of Web Log Server Records of Murdoch University before Implementation of Proposed Data Cleaning Algorithm
From Table 5 it is clearly evident that the modified UUI algorithm is far better than the older algorithm and algorithms proposed by Shetal A. Raiyani in their work. The modified algorithm shows much clarity in data cleaning and also proves in its efficiency by the consuming less execution time. It takes only 0.3582 second to identify the number of unique users for the data size of $10^{12}$ records, whereas the referred algorithm consumes 0.2567 and 0.5432 seconds for data sizes of $10^3$ and $10^4$ seconds. For the data size of $10^3$ record sizes, UUI algorithm proposed by K. R. Sangeetha and Dr. Krishnamurthy consume 0.5432 seconds to identify 1765 unique users whereas the proposed modified UUI algorithm takes 0.3724 seconds to identify 44675422 unique users, which proves that the modified UUI algorithm takes less time to execute despite the huge data size. Still work is in progress to fine tune the algorithm and improve its efficiency to an appreciable extent.

From the above Figures 8 and 9, it is evident that the proposed Data Cleaning algorithm has performed well. Sample of 642 records are taken from Murdoch University Web Log Server and the data are cleaned using the proposed Data Cleaning algorithm, interestingly 342 irrelevant records were eliminated at a time factor of 1.25 (s). This result is a valid proof for the performance of the proposed Data Cleaning Algorithm.\[10\] [11].
7. Conclusion

This paper has come out with a modified version of UUI algorithm with modified coalesced hashing technique to identify unique users. The older version of UUI algorithm proposed lacked in accuracy to some extent and efficiency to some extent. To resolve this hashing technique proposed in previous study has been modified by inclusion coalescing strategy to reduce storage and remove the collision problems that existed in the previously proposed hashing methodologies. The results obtained from the experimental analysis clearly proves that the modified UUI Algorithm along with coalesced hashing technique has produced promising results over the previous study in terms of accuracy and efficiency. Various evaluation and comparisons are made to prove the improved performance of the algorithm. The results obtained prove the verdict.

This paper has also comes out with a unique strategy to group the IP addresses according to their zone specification. From the grouped IP address, this work uses the Binary Search technique to locate the IP of the unique user. Inclusion of this strategy in the previously proposed DUI algorithm drastically minimizes the time to search and locate users IP addresses. The algorithm is evaluated with different universities web log server’s data to identify the efficiency of the cleaning process, to check the number of users visited the pages, time taken to identify the unique users etc. The algorithm proves and shows much improvement over the previous and other related works. Further improvements are needed to combine the whole process of Web Usage Mining. A complete methodology that covers pattern discovery and pattern analysis will be more useful in the user identification process. This work helped the site developers to analyze their sites and also helped them to identify the user types and range of users. It also guided them further for redesigning their sites according to the users requirements.

References


<table>
<thead>
<tr>
<th></th>
<th>Reference</th>
</tr>
</thead>
</table>